UNIVERSITÀ DEGLI STUDI DI PALERMO

DEPARTMENT	Ingegneria
ACADEMIC YEAR	2021/2022
BACHELOR'S DEGREE (BSC)	COMPUTER ENGINEERING
SUBJECT	ALGORITHMS AND DATA STRUCTURES
TYPE OF EDUCATIONAL ACTIVITY	B
AMBIT	50289-Ingegneria informatica
CODE	01175
SCIENTIFIC SECTOR(S)	ING-INF/05
HEAD PROFESSOR(S)	LO PRESTI LILIANA Professore Associato Univ. di PALERMO
OTHER PROFESSOR(S)	
CREDITS	9
INDIVIDUAL STUDY (Hrs)	144
COURSE ACTIVITY (Hrs)	81
PROPAEDEUTICAL SUBJECTS	
MUTUALIZATION	
YEAR	2
TERM (SEMESTER)	1° semester
ATTENDANCE	Not mandatory
EVALUATION	Out of 30
TEACHER OFFICE HOURS	**LO PRESTI LILIANA** Tuesday 16:00 17:00 Luogo da concordare via email col docente. Preferibilmente a distanza su piattaforma MS Teams.

DOCENTE: Prof.ssa LILIANA LO PRESTI

PREREQUISITES	Good Knowledge of programming principles (C or Java languages)
LEARNING OUTCOMES	- Knowledge and understanding At the end of the course, the student will have knowledge of the problems concerning algorithms and data structures. In particular, the student will know in detail the algorithm analysis methodologies, the elementary data structures, the main search and sorting algorithms and basic knowledge related to graphs. To achieve this goal the course includes: lectures; analysis and discussion of program fragments. - Ability to apply knowledge and understanding The student will be able to evaluate the characteristics, advantages and limitations of the main algorithms and data structures. He will be able to design, analyze and assess solutions to medium complexity problems. He will also be able to develop new software solutions, assessing their quality in terms of simplicity, effectiveness and efficiency. To achieve this goal the course includes exercises to be performed in the classroom related to the analysis of code fragments / pseudo-code, and to the design of efficient algorithms. To verify this objective the exam will include the analysis of existing algorithms, the application of known techniques and the design of an algorithm based on the textual description of the problem to be solved. - Autonomy of judgment The student will be able both to carry out the analysis of a problem and to design a suitable software solution. He will be able to evaluate the quality of a software solution in terms of simplicity, readability, efficiency and possibility of reuse. To achieve this goal the course includes: analysis and discussion of code fragments / pseudo-code; lessons related to algorithm analysis techniques; discussions on possible advantages and disadvantages deriving from the use of particular data structures. The student will be initially encouraged to find and independently assess solutions to the problems, in order to be able to understand the quality and usefulness of the solutions proposed during the course. For the verification of this objective the exam will include the design of an algorithm. The student will have to make design choices independently, such as, for example, the choice of the data structure suitable for the implementation of an efficient solution. - Communication skills The student will acquire the ability to communicate and express problems concerning the topics of the course. He will be able to discuss issues related to the software implementation of algorithms and efficient data structures. To achieve this goal the course will include classroom exercises during which students will propose solutions to the proposed exercises and discuss any difficulties encountered. To verify this objective, the exam will include an oral discussion on the topics covered in the course. - Learning skills The student will have to develop the ability to learn the analysis and synthesis processes related to the coding of algorithms of medium complexity and to the relative implementation of libraries and software tools. To achieve this goal the course includes: exercises to be performed independently; discussion of any difficulties encountered. For the verification of this objective the exam will include the discussion of topics introduced in class and that students should elaborate on independently.
ASSESSMENT METHODS	Students will have to take a written and an oral test. The written test includes exercises aiming at ascertaining the candidate's ability to analyze the computational cost of an algorithm, understand algorithms, design an algorithm using the programming techniques learned during the course, apply the algorithms studied during the course. Candidates who reach sufficiency (18/30) in the written test are admitted to take the oral exam. The oral test consists of the discussion of the written test and questions aiming at ascertain the candidate's knowledge of the topics covered in the course. The final mark, in 18/30 - 30/30 with honors, is calculated as average of the overall evaluation of the written and oral exams. According to the Dublin descriptors, the expected results will be assessed in relation to the final grade as follows: - from 18/30 to 20/30: mediocre or sufficient knowledge and understanding of the topics covered, partial ability to apply the acquired knowledge to solve the proposed problems; partial autonomy of judgment, communication skills and the

	ability to learn. - from 21/30 to 23/30: sufficient or discrete knowledge and understanding of the topics covered, sufficient ability to apply the knowledge acquired for the resolution of the proposed problems, sufficient independence of judgment, communication skills and ability to learn. - from 24/30 to 26/30: discrete knowledge and understanding of the topics covered, discrete ability to apply the knowledge acquired for the resolution of the proposed problems, sufficient autonomy of judgment, communication skills and the ability to learn. - from 27/30 to 30/30 cum laude: good or excellent knowledge and understanding of the topics covered, good or excellent ability to apply the acquired knowledge for the resolution of the proposed problems, good or excellent judgment autonomy, enables communicative and ability to learn Minimum requirement for passing the exam is the demonstrated knowledge, in the two tests, of the basic notions related to: analysis and understanding of algorithms, calculation of the complexity of an algorithm, management of elementary data structures (arrays, stacks, queues, trees and hash tables) and complex structures (priority queues, union-find and graphs), sorting and searching algorithms.
EDUCATIONAL OBJECTIVES	The course will deal in-depth with the study of algorithms and data structures. The student will acquire a good knowledge of the aspects related to the analysis of the efficiency of an algorithm, specific algorithms of search and data sorting, the implementation of data structures such as stacks, queues, trees, hash tables and priority queues. The course will aim to highlight the advantages and disadvantages of the various available techniques based on the problem to be solved. The course aims to introduce concepts related to particular programming techniques such as divide and conquer, dynamic programming and greedy strategies. Finally, the course will introduce problems and algorithms related to graphs and basic concepts about numerical optimization.
TEACHING METHODS	Lectures and exercises
SUGGESTED BIBLIOGRAPHY	C. Demetrescu, I. Finocchi, G. F. Italiano (2008). Algoritmi e strutture dati - seconda edizione. McGraw-Hill T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, L. Colussi (2010). Introduzione agli algoritmi e strutture dati - terza edizione. McGraw-Hill

SYLLABUS

Hrs	Frontal teaching
6	Introduction to algorithms. Iterative and recursive algorithms. Memory occupancy and algorithm efficiency. Algorithm analysis. The asymptotic notation big-O, big-Omega, big-Theta. Analysis of the worst, best and average cases. Analysis of recursive algorithms.
6	Definition of data structure; typical operations for the management of data structures (insertion, deletion and search); indexed and linked representations; arrays and their resizing; linked structures: records and pointers; simple list, doubly linked list, circular list; batteries and queues; trees: basic definitions; representation techniques of trees in memory; tree visit algorithms and their computational complexity.
8	The problem of sorting n objects; decision tree and height; lower-bound of sorting algorithms in the worst case; sorting algorithms with quadratic upper-bound: insertionSort, selectionSort, bubbleSort; sorting in n log n: heapSort, mergeSort and quickSort; definition of heap and its properties; local ordering; randomization; sorting integers and objects with integer keys: integerSort, bucketSort, radixSort.
5	Search binary trees and search operations, insertion and deletion of elements; AVL trees and balancing factor, Fibonacci trees, insertion and deletion of elements in AVL trees, SS, DS, SD, DD rotation operations; analysis of algorithm complexity; Red-Black trees
4	Direct access tables and load factor; hash tables; collision definition; perfect and non-perfect hash functions; uniformity of hash functions; construction of hash functions: division method, folding method, multiplication method, polynomial approach; collision resolution: collision lists and open addressing, linear scanning method and quadratic scanning, double hashing; cost analysis for hash tables.
4	Priority queues, insertion and deletion operations, d-heap and its characteristics; binomial trees and their properties; binomial heap; restructuring of a binomial heap; cost analysis on binomial heaps.
2	Definition of union-find and operations of interest; quickFind and quickUnion; balancing heuristics of union operations; balancing heuristics for the quickUnion; union by rank and union by size; compression heuristics in the find operation; computational cost analysis for union-find with and without heuristics.
5	Algorithmic techniques: divide et impera; dynamic programming; greedy strategies; dynamic programming applications: sub-array of maximum value; paths of minimum value in a matrix; distance between strings (edit distance); maximum common sub-sequence.
12	Graphs and representation of graphs in memory: list of arcs, lists of adjacency, lists of incidence, matrices of adjacency, matrices of incidence; graph visit algorithms; connected graphs and finding connected components in undirected graph; strong connectivity in oriented graphs and strongly connected components in oriented graphs; definition of minimum spanning tree; Prim and Kruskal algorithms; definition of minimum path; distance between nodes; Bellman-Ford algorithm and Dijkstra algorithm.
4	Theory of NP-completeness: notes on non-deterministic machines; complexity classes: problems in P, NP, NP-C and NP- H. Examples.

SYLLABUS

Hrs	Frontal teaching
4	Introduction to numerical optimization. Introduction to convex problems and to numerical optimization problems (interior-point method). Introduction to gradient-descent methods.

Hrs	Practice
4	Estimation of computational cost and memory occupation of algorithms. Identification of the complexity class of an algorithm.
4	Testing data structures: binary search trees; hash tables, AVL trees, priority queues.
8	Design of algorithms for solving problems of medium complexity.
5	Simulation of algorithms on graphs.